Plotting and Visualization

- One important task in Explorative Data Analysis (EDA) is to plot and visualize data.
- Python provides several packages to plot data.
- The matplotlib package is one of those packages. It is very similar to the plotting functions provided by MatLab and R.
- The pandas package also come with plotting functions that are based on matplotlib
- Another package is seaborn that plot statistical features of the data

We introduce the basic plotting features here.

In [ ]:
```
%matplotlib inline
from __future__ import division
from numpy.random import randn
import numpy as np
np.random.seed(12345)
np.set_printoptions(precision=4)
import pandas as pd
from pandas import Series, DataFrame
import os
```

In [ ]:
```
%pwd
```

A brief matplotlib API primer

The Pyplot API

The matplotlib.pyplot (https://matplotlib.org/devdocs/api/pyplot_summary.html) module contains functions to generate many kinds of plots quickly. For examples that showcase the use of the matplotlib.pyplot module, see the Pyplot tutorial (../tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py) or the Pyplot Examples (../gallery/index.html#pyplots-examples).

In [ ]:
```
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 6))
```

In [ ]:
```
# Plot 50 random numbers, x-axis is a range, y-axis is the value, dashline, black
plt.plot(randn(50).cumsum(), 'k--')
```

See matplotlib.pyplot.plot (https://matplotlib.org/devdocs/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot) for details

Figures and Subplots

- Figure Class (https://matplotlib.org/devdocs/api/_as_gen/matplotlib.figure.Figure.html#matplotlib.figure.Figure)
- Axes Class (https://matplotlib.org/api/axes_api.html)

- A figure is an object that can contain one or more subplots.
- The subplots are organized into grid.
- Each subplot is represented by an Axes object, which provides functions to draw the subplot

In [ ]:
```
# Create a fig
fig = plt.figure()
# Add 1st subplot on a grid with 2x2 cell per plot
# Create an Axes object
ax1 = fig.add_subplot(2, 2, 1)
fig
```

In [ ]:
```
# Add two more subplots
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
fig
```

Axes Objects

Axes object has functions to produce many different type of plots:

- Basic
- Spans
- Spectral
- Statistics
- Binned
- Contours
- Array
- Unstructured Triangles
- Text and Annotations
- Fields

In [ ]:
```
# plot in sub-figure 1 the histogram of 100 random numbers in 20 bins, semi-transp
arent
_=ax1.hist(randn(100), bins=20, color='k', alpha=0.3)
# plot a scatter plot of 30 randomly generated 2-d points
ax2.scatter(np.arange(30), np.arange(30) + 3 * randn(30))
# plot a dashed line of 50 random numbers
ax3.plot(randn(50).cumsum(), 'k--')
fig
```

In [ ]:
```
plt.close('all')
```

In [ ]:
```
# Use subplots() to create a fig with 6 sub-figures
fig, axes = plt.subplots(2, 3)
axes
```

Adjusting the spacing around subplots

subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)

The parameter meanings (and suggested defaults) are:

- left = 0.125 # the left side of the subplots of the figure
- ight = 0.9 # the right side of the subplots of the figure
- bottom = 0.1 # the bottom of the subplots of the figure
- top = 0.9 # the top of the subplots of the figure
- wspace = 0.2 # the amount of width reserved for blank space between subplots, expressed as a fraction of the average axis width
- hspace = 0.2 # the amount of height reserved for white space between subplots, expressed as a fraction of the average axis height

In [ ]:
```
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
# plat histograms in each axis
for i in range(2):
    for j in range(2):
        axes[i, j].hist(randn(500), bins=50, color='k', alpha=0.5)
plt.subplots_adjust(wspace=0, hspace=0)
```

Colors, markers, and line styles

The plot function can take parameters that describe the colors, markers, and styles of various components of a plot. The descriptions is given as predefined strings.

character description

- '-' solid line style
- '--' dashed line style
- '-.' dash-dot line style
- ':' dotted line style
- '.' point marker
- ',' pixel marker
- 'o' circle marker
- 'v' triangle_down marker
- '^' triangle_up marker
- '<' triangle_left marker
- '>' triangle_right marker
- '1' tri_down marker
- '2' tri_up marker
- '3' tri_left marker
- '4' tri_right marker
- 's' square marker
- 'p' pentagon marker
- '*' star marker
- 'h' hexagon1 marker
- 'H' hexagon2 marker
- '+' plus marker
- 'x' x marker
- 'D' diamond marker
- 'd' thin_diamond marker
- '|' vline marker
- '_' hline marker

The following color abbreviations are supported:

- ‘b’ blue
- ‘g’ green
- ‘r’ red
- ‘c’ cyan
- ‘m’ magenta
- ‘y’ yellow
- ‘k’ black
- ‘w’ white

In [ ]:
```
# Create a figure
plt.figure()
```

In [ ]:
```
plt.plot(randn(30).cumsum(), 'ko--')
```

In [ ]:
```
plt.close('all')
```

In [ ]:
```
# Plot multiple lines in one figure
data = randn(30).cumsum()
plt.plot(data, 'g--', label='Default')

# drawstyle= [‘default’ | ‘steps’ | ‘steps-pre’ | ‘steps-mid’ | ‘steps-post’]
plt.plot(data, 'b-', drawstyle='steps-post', label='steps-post')
plt.legend(loc='best')
```

See matplotlib.pyplot.plot (https://matplotlib.org/devdocs/api/_as_gen/matplotlib.pyplot.plot.html#matplotlib.pyplot.plot) for details

Ticks, labels, and legends

Setting the title, axis labels, ticks, and ticklabels

Each Axes object can define ticks for x and y axises. The ticks can be set on specific positions, with its own label. The label font size and direction can also be set.

```
In [ ]: fig = plt.figure()
        # use an Axes object to contol the ploting features
        ax = fig.add_subplot(1, 1, 1)

        ticks = ax.set_xticks([0, 250, 500, 750, 1000])
        labels = ax.set_xticklabels(['one', 'two', 'three', 'four', 'five'],
                                    rotation=30, fontsize='small')
        ax.set_title('A plot of cumsum for 1000 random numbers')
        ax.set_xlabel('Stages')

        ax.plot(randn(1000).cumsum())
```

Adding legends

```
In [ ]: fig = plt.figure(); ax = fig.add_subplot(1, 1, 1)
        # plot three lines from different sets of data
        ax.plot(randn(1000).cumsum(), 'k', label='one')
        ax.plot(randn(1000).cumsum(), 'g--', label='two')
        ax.plot(randn(1000).cumsum(), 'r.', label='three')

        ax.legend(loc='best')
```

Annotations and drawing on a subplot

When a curve is plotted, the user may add notes and markers to the figure to indicate interesting points on the curve

In [ ]:
```
from datetime import datetime

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

# read SP500 index data into a DataFrame
data = pd.read_csv('ch08/spx.csv', index_col=0, parse_dates=True)
spx = data['SPX']
print("data =\n", data[:5])
print("spx =\n", spx[:5])

# Plot SP500 index data
spx.plot(ax=ax, style='k-')

crisis_data = [
    (datetime(2007, 10, 11), 'Peak of bull market'),
    (datetime(2008, 3, 12), 'Bear Stearns Fails'),
    (datetime(2008, 9, 15), 'Lehman Bankruptcy')
]

for date, label in crisis_data:
    # add annotation in the plot, set coordinates for text and arrow
    ax.annotate(label, xy=(date, spx.asof(date) + 50),
                xytext=(date, spx.asof(date) + 200),
                arrowprops=dict(facecolor='black'),
                horizontalalignment='left', verticalalignment='top')

# Zoom in on 2007-2010
ax.set_xlim(['1/1/2007', '1/1/2011'])
ax.set_ylim([600, 1800])

ax.set_title('Important dates in 2008-2009 financial crisis')
```

In [ ]:
```
# Draw shapes in a plot
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

rect = plt.Rectangle((0.2, 0.75), 0.4, 0.15, color='k', alpha=0.3)
circ = plt.Circle((0.7, 0.2), 0.15, color='b', alpha=0.3)
pgon = plt.Polygon([[0.15, 0.15], [0.35, 0.4], [0.2, 0.6]],
                   color='g', alpha=0.5)

ax.add_patch(rect)
ax.add_patch(circ)
ax.add_patch(pgon)
```

Saving plots to file

Plotted figures can be saved into an image file and read back in later

In [ ]:
```
fig
```

In [ ]:
```
fig.savefig('figpath.svg')
```

In [ ]:
```
fig.savefig('figpath.png', dpi=100, bbox_inches='tight')
```

```
In [ ]: # Save plot into an in-memory file
        from io import BytesIO
        buffer = BytesIO()
        plt.savefig(buffer, format='png')
        plot_data = buffer.getvalue()
```

```
In [ ]: plot_data
```

Read and Display images from image files

Need to use packages for image read and display

```
In [ ]: import matplotlib.image as mpimg
        # read image into a NUmPy narray
        img = mpimg.imread('figpath.png')
        plt.imshow(img)
```

```
In [ ]: from PIL import Image
        buffer.seek(0)
        im = Image.open(buffer)
        print(im.format, im.size, im.mode)
        # Need to install XV in order to show the image correctly
        #im.show()
        plt.imshow(im)
```

matplotlib configuration

The configuration of current figure can be modified with the plt.rc() function

```
In [ ]: plt.rc('figure', figsize=(10, 10))
        plt.plot(randn(50).cumsum(), 'k--')
```

```
In [ ]: plt.rc('figure', figsize=(2, 2))
        plt.plot(randn(50).cumsum(), 'k--')
```

```
In [ ]: plt.rc('figure', figsize=(6, 6))
```

Plotting functions in pandas

- The objects in pandas, Series and DataFrame, come with a plot() function to faciliate plotting
- Also, Series or columns in DataFrames can be passed into matplotlib functions to generate plots

Line plots

```
In [ ]: plt.close('all')
```

```
In [ ]: s = Series(np.random.randn(10).cumsum(), index=np.arange(0, 100, 10))
        s.plot()
```

```
In [ ]: df = DataFrame(np.random.randn(10, 4).cumsum(0),
                       columns=['A', 'B', 'C', 'D'],
                       index=np.arange(0, 100, 10))
        df.plot()
```

Bar plots

```
In [ ]: fig, axes = plt.subplots(2, 1)
        data = Series(np.random.rand(16), index=list('abcdefghijklmnop'))
        data.plot(kind='bar', ax=axes[0], color='k', alpha=0.7)
        data.plot(kind='barh', ax=axes[1], color='k', alpha=0.7)
```

```
In [ ]: df = DataFrame(np.random.rand(6, 4),
                       index=['one', 'two', 'three', 'four', 'five', 'six'],
                       columns=pd.Index(['A', 'B', 'C', 'D'], name='Genus'))
        df
        df.plot(kind='bar')
```

```
In [ ]: plt.figure()
```

```
In [ ]: df.plot(kind='barh', stacked=True, alpha=0.5)
```

```
In [ ]: tips = pd.read_csv('ch08/tips.csv')
        print(tips)
        #party_counts = pd.crosstab(tips.day, tips.size)
        party_counts = pd.crosstab(tips['day'], tips['size'])
        party_counts
```

```
In [ ]: # Not many 1- and 6-person parties
        party_counts = party_counts.loc[:, 2:5]
        party_counts
```

```
In [ ]: # Normalize to sum to 1
        party_pcts = party_counts.div(party_counts.sum(1).astype(float), axis=0)
        party_pcts
```

```
In [ ]: party_pcts.plot.bar()
```

Histograms and density plots

The obj.hist() function for pandas objects is used to plot histograms.

```
In [ ]: plt.figure()
```

```
In [ ]: tips['tip_pct'] = tips['tip'] / tips['total_bill']
        tips['tip_pct'].hist(bins=50)
```

```
In [ ]: plt.figure()
```

```
In [ ]: tips['tip_pct'].plot(kind='kde')
```

In [ ]:
```
plt.figure()
```

In [ ]:
```
comp1 = np.random.normal(0, 1, size=200)  # N(0, 1)
comp2 = np.random.normal(10, 2, size=200)  # N(10, 4)
values = Series(np.concatenate([comp1, comp2]))
values.hist(bins=100, alpha=0.3, color='k', normed=True)
values.plot(kind='kde', style='k--')
```

Scatter plots

In [ ]:
```
macro = pd.read_csv('ch08/macrodata.csv')
data = macro[['cpi', 'm1', 'tbilrate', 'unemp']]
trans_data = np.log(data).diff().dropna()
trans_data[-5:]
```

In [ ]:
```
plt.figure()
```

In [ ]:
```
plt.scatter(trans_data['m1'], trans_data['unemp'])
plt.title('Changes in log %s vs. log %s' % ('m1', 'unemp'))
```

In [ ]:
```
#pd.scatter_matrix(trans_data, diagonal='kde', color='k', alpha=0.3)
pd.scatter_matrix(trans_data, diagonal='kde', alpha=0.3)
```

Visualize Geographic Data on Maps

- Matplotlib uses Basemap object to visualize geographic data. The Basemap class is within mpl_toolkits package
- If Basemap is not found, run the following command in a terminal window:
  - conda install -c conda-forge basemap basemap-data-hires
- A Basemap object maintains information of a map, including types of earth project, direction, area, distance, shape, latitute, longitive, etc.
- The Basemap package contains a range of useful functions for drawing borders of physical features like continents, oceans, lakes, and rivers, as well as political boundaries such as countries and US states and counties.
- Provide map-specific methods to place data on map:
  - contour()/contourf() : Draw contour lines or filled contours
  - imshow(): Draw an image
  - pcolor()/pcolormesh() : Draw a pseudocolor plot for irregular/regular meshes
  - plot(): Draw lines and/or markers.
  - scatter(): Draw points with markers.
  - quiver(): Draw vectors.
  - barbs(): Draw wind barbs.
  - drawgreatcircle(): Draw a great circle.

Plotting Maps: Visualizing Haiti Earthquake Crisis data

- The data is in a csv file and the map is in data files from basemap package

In [ ]:
```
data = pd.read_csv('ch08/Haiti.csv')
data.info()
```

In [ ]:
```
data[['INCIDENT DATE', 'LATITUDE', 'LONGITUDE']][:10]
```

In [ ]:
```
data['CATEGORY'][:6]
```

In [ ]:
```
data.describe()
```

In [ ]:
```
data = data[(data.LATITUDE > 18) & (data.LATITUDE < 20) &
            (data.LONGITUDE > -75) & (data.LONGITUDE < -70)
            & data.CATEGORY.notnull()]
data[:][:10]
```

In [ ]:
```
def to_cat_list(catstr):
    stripped = (x.strip() for x in catstr.split(','))
    return [x for x in stripped if x]

def get_all_categories(cat_series):
    cat_sets = (set(to_cat_list(x)) for x in cat_series)
    return sorted(set.union(*cat_sets))

def get_english(cat):
    code, names = cat.split('.')
    if '|' in names:
        names = names.split(' | ')[1]
    return code, names.strip()
```

In [ ]:
```
get_english('2. Urgences logistiques | Vital Lines')
```

In [ ]:
```
all_cats = get_all_categories(data.CATEGORY)
# Generator expression
english_mapping = dict(get_english(x) for x in all_cats)
english_mapping['2a']
english_mapping['6c']
```

In [ ]:
```
def get_code(seq):
    return [x.split('.')[0] for x in seq if x]

all_codes = get_code(all_cats)
code_index = pd.Index(np.unique(all_codes))
dummy_frame = DataFrame(np.zeros((len(data), len(code_index))),
                        index=data.index, columns=code_index)
```

In [ ]:
```
dummy_frame.iloc[:, :6].info()
```

In [ ]:
```
for row, cat in zip(data.index, data.CATEGORY):
    codes = get_code(to_cat_list(cat))
    dummy_frame.ix[row, codes] = 1

data = data.join(dummy_frame.add_prefix('category_'))
```

In [ ]:
```
data.iloc[:, 10:15].info()
```

This will take some time to draw ...

In [ ]:
```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt

def basic_haiti_map(ax=None, lllat=17.25, urlat=20.25,
                    lllon=-75, urlon=-71):
    # create polar stereographic Basemap instance.
    m = Basemap(ax=ax, projection='stere',
                lon_0=(urlon + lllon) / 2,
                lat_0=(urlat + lllat) / 2,
                llcrnrlat=lllat, urcrnrlat=urlat,
                llcrnrlon=lllon, urcrnrlon=urlon,
                resolution='f')
    # draw coastlines, state and country boundaries, edge of map.
    m.drawcoastlines()
    m.drawstates()
    m.drawcountries()

    return m

p = basic_haiti_map()
p.plot(100, 100, 'k.', alpha=0.5)
```

In [ ]:
```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 10))
fig.subplots_adjust(hspace=0.05, wspace=0.05)

to_plot = ['2a', '1', '3c', '7a']

lllat=17.25; urlat=20.25; lllon=-75; urlon=-71

for code, ax in zip(to_plot, axes.flat):
    m = basic_haiti_map(ax, lllat=lllat, urlat=urlat,
                        lllon=lllon, urlon=urlon)

    cat_data = data[data['category_%s' % code] == 1]

    # compute map proj coordinates.
    x, y = m(cat_data.LONGITUDE.values, cat_data.LATITUDE.values)

    m.plot(x, y, 'k.', alpha=0.5)
    ax.set_title('%s: %s' % (code, english_mapping[code]))
```

In [ ]:
```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 10))
fig.subplots_adjust(hspace=0.05, wspace=0.05)

to_plot = ['2a', '1', '3c', '7a']

lllat=17.25; urlat=20.25; lllon=-75; urlon=-71

def make_plot():

    for i, code in enumerate(to_plot):
        cat_data = data[data['category_%s' % code] == 1]
        lons, lats = cat_data.LONGITUDE, cat_data.LATITUDE

        ax = axes.flat[i]
        m = basic_haiti_map(ax, lllat=lllat, urlat=urlat,
                            lllon=lllon, urlon=urlon)

        # compute map proj coordinates.
        x, y = m(lons.values, lats.values)

        m.plot(x, y, 'k.', alpha=0.5)
        ax.set_title('%s: %s' % (code, english_mapping[code]))


make_plot()
```

In [ ]:
```
shapefile_path = 'ch08/PortAuPrince_Roads/PortAuPrince_Roads'
m.readshapefile(shapefile_path, 'roads')
```